UIUCDCS-R-80-1035

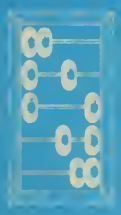
UILU-ENG 80 1734

ILLINET--A 32 Mbits/sec. Local Area Network*

by

W.Y. Cheng
S. Ray
R. Kolstad
J. Luhukay
R. Campbell
J.W-S. Liu

October, 1980



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

ILLINET--A 32 Mbits/sec. Local-Area Network*

W.Y. Cheng
S. Ray
R. Kolstad
J. Luhukay
R. Campbell
J.W-S. Liu

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

*Supported in part by grant NSF MCS79-06945.

Abstract

ILLINET is a fiber-optical ring network designed to provide wide band linkages between host computers for the purpose of facilitating file transfers at speeds near those of fast I/O devices in the hosts. Its structure is similar to the Distributed Computing System. ILLINET will eventually connect several PDP-11's, a PRIME computer and a network of microcomputers. These computers are used in a variety of real-time and batch processing applications. Currently they are already interconnected via 9600 band lines in a star configuration to provide access to simple terminals. This paper describes the network architecture, control structure, and hardware configuration of ILLINET.

1. Introduction

The rapid development in VLSI technology has made host computers and terminals smaller and cheaper. In recent years it has become rather common for an organization to have several computing systems with substantial processing and memory capacity operated and maintained within the same building or in several closely located buildings. These computing systems may each serve a wide range of simple and intelligent terminals. The need to share data, programs, processing power, and I/O facilities invariably makes it necessary for the computers and terminals to be interconnected in the form of a local area network. Indeed, many local area networks have been designed and implemented. Among the well known local area networks are Xerox ETHERNET [1], Bell SPIDER [2], and LCSNET [3]. These networks have been designed to provide low delay access via interactive terminals to host computers at relatively low cost per interconnection and with ease for network extension and reconfiguration. Since the effective link bandwidth in such a network is divided evenly among all terminals and hosts, it is often impossible to facilitate transfer of large files between host computers at high speeds required in many applications.

Many studies have shown that the performance of a local resource sharing network and distributed data base system depends critically on the communication bandwidth between hosts [4]. In particular, an effective resource sharing environment can be achieved only when wide

band data links between hosts are available to allow file transfers at speeds near those of fast I/O devices in the hosts. ILLINET is a local area network designed to accomplish this goal. Its structure is similar to the Distributed Computing System (DCS) at the University of California, Irvine [5]. This paper describes ILLINET which has been designed and is currently being implemented in the Department of Computer Science at the University of Illinois at Urbana-Champaign.

In section 2 the design objectives of ILLINET are discussed. These objectives impose several constraints on the network configuration and control structure. Section 3 gives an overview of ILLINET. In section 4 the link level data packet format is described together with the hardwired network access and link control protocols. Finally, the hardware architecture is presented in section 5.

2. Design Objectives

ILLINET will eventually connect several PDP-11's, a PRIME computer, and a network of microcomputers. These computers are operated and maintained by the Department and are used in a variety of real-time and batch processing applications. Currently they are already interconnected via 9600 baud lines in a star configuration to provide access to 50-60 simple terminals. All of these computers are located within one building although ILLINET is designed to allow interbuilding connections. It is envisioned that one of the nodes on ILLINET will be

a PDP-11 which will serve as a gateway to the main campus computing facility.

The primary purpose for the design and implementation of ILLINET is to enhance existing computation facilities so that the resultant computer network will support effectively a variety of research activities in the areas of distributed operating systems, distributed data base systems and file servers. In order to assure that transmission links between the nodes and link-control level protocols will not be the bottleneck in interprocessor communication and data flow, it was decided that ILLINET is to be constructed using the latest cost effective technology. The transmission medium used in ILLINET is fiber optics because of its ability to support high bit rates and allow reasonable interfaces. A link bandwidth of 32 Mbits per second is achieved with the use of ECL circuits. Since most of the network access and link control protocol functions are implemented in hardware, nearly all this bandwidth will be available for interprocessor communication.

The need to avoid the difficult task of providing bidirectional signal transmission and proper termination of the optical fibers dictated that ILLINET be a ring network. The packet switching discipline and distributed network control structure are used. Because of the high data link bandwidth and the relative short loop delay in ILLINET, it is not necessary that the most efficient network access control scheme be used. The version of token control scheme implemented

in ILLINET is described in sections 3 and 5. It is similar to the scheme used in DCS. It will undoubtedly provide sufficiently low access delay and high network throughput.

In order to support high-level process communication in broadcast mode and to allow transparent transfer of destination process from one node to another, associative addressing is used in ILLINET. Address recognition hardware and link control protocols are both designed to support efficient broadcast communication in the network.

3. Network Overview

The configuration of ILLINET is described in Figure 1. It contains no central controller or primary station to carry out clock synchronization and access control functions. In each of the ring adaptors (RA) on the ring, there is a 16-bit active data path (hereafter referred to as front-end window) between the optical receiver and transmitter in the front end. More specifically, a RA functions as a repeater which retransmits the incoming data stream. The portion of the data stream appearing in the front-end window may be examined by the RA.

A host can gain access to the network via the ring adaptor attached to it. To each of the hosts on ILLINET, the network functions as a packet-switched network. To send a message, the host segments the message into network packets of a maximum size of 4K bits. Each packet is delivered individually to the RA where it is stored in one of the

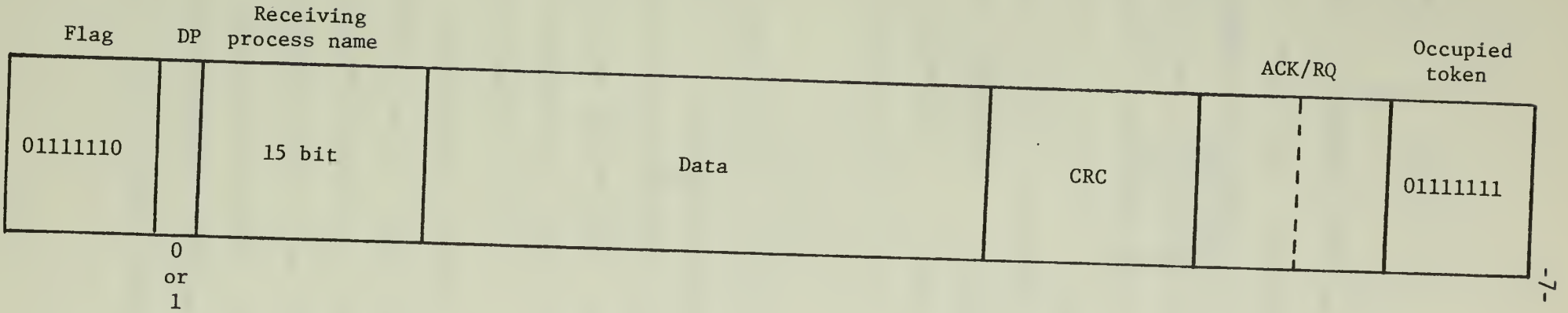

Flag
DP
Receiving process name
ACK/RQ
Occupied token
01111110
15 bit
Data
CRC
01111111
0 or 1

Figure 2

output buffers. The completion of the loading of the data packet into the output buffer is acknowledged by an interrupt sent by the RA to the host. The host in turn can signal the RA to commence accessing the network and transmitting the data packet. The transmission of the data packet is then carried out under the control of the RA without host intervention. Under normal operating conditions, the data packets will be delivered to the destination in the order in which they are sent from the host to the RA, and duplicate and lost packets will not occur. However, reliable sequenced delivery is not guaranteed. Mechanisms to assure reliable datagram delivery and message sequencing and reassembly are carried out by the hosts.

The RA monitors the data stream passing through its front-end window at all times. When there is a packet to be delivered, the RA removes the access control token (01111111) from the ring when the token appears at its front-end window. There is only one control token in the ring. When the RA receives the token, it is allowed to transmit one data packet. The format of the data packet is shown in Figure 2. Besides the receiving process name there are CRC check, duplicate mark and acknowledge/repeat request fields. The data packet is retransmitted until positive acknowledgements are received from all RA's serving active processes whose names match the receiving process name in the packet header. (We will return to discuss the acknowledgment and repeat request features in the next section.) The use of this stop-and-wait ARQ scheme simplifies the host-to-host synchronization. Since the bandwidth

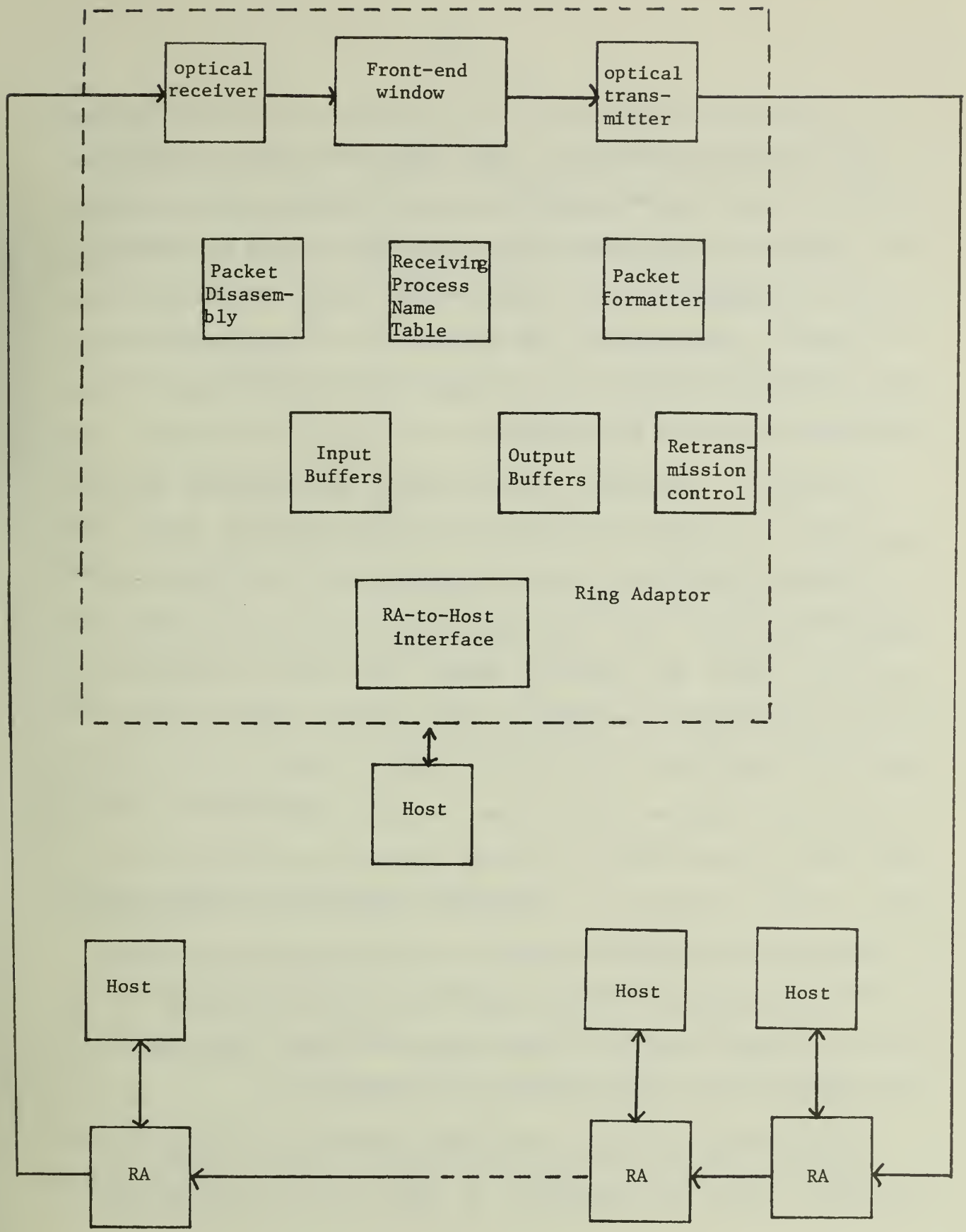

optical receiver
Front-end window
optical trans-mitter
Packet Disasem-bly
Receiving Process Name Table
Packet formatter
Input Buffers
Output Buffers
Retrans-mission control
RA-to-Host interface
Ring Adaptor
Host
Host
Host
Host
RA
RA
RA

Figure 1

of the host-to-RA interface is significantly lower than the network bandwidth, the host-to-host throughput will not be limited by its use [6]. Furthermore, since there are two output buffers in the RA, network access for transmission from one buffer over the network can be carried out while the host loads the other output buffer. Thus, the speed of large file transfer between hosts will be limited primarily by the bandwidth of the host-to-RA interfaces.

In order to facilitate dynamic renaming and broadcast mode communication, a high-speed static RAM is provided in each RA to store the local active process name table as suggested in [7]. This table is updated by the host. When a data packet passes the front-end window of a RA, the RA checks the receiving process name field to determine whether it matches any of the names in its process name table. The data packet is copied into one of the 16 input buffers as it is simultaneously subjected to CRC checks. The data packet is kept in the input buffer only when there is a process name match, the data packet is free of error, and there is an input buffer available for its storage. In this case, the RA interrupts the host to inform it of the reception of the data packet. As the data packet passes through its front-end window, the RA makes comment in the acknowledgment/repeat request field. Such comments serve as acknowledgments to the sending RA.

Within the ring only the data path between the optical receiver and transmitter inside the sending RA is open. Hence, under normal

operating conditions, the sending RA will remove the data packet when it returns to the optical receiver. By checking the contents of the acknowledgment/repeat request field in the returned data packet, the sending RA can decide immediately whether retransmission of the data packet is warranted. Either when the data packet transmission is completed successfully or is aborted after retransmission a maximum number of times, the sending RA releases the token and interrupts the host. By checking the status of the RA, the sending host can determine whether the transmission of the data packet is successful. If the other output buffer is nonempty and if the transmission of the previous data packet is successful, the host may signal the RA to commence network access again. On the other hand, if the delivery of the data packet fails, the host may ask the RA to attempt retransmission again or to invoke error diagnosis process. Thus, the sending RA is guaranteed the use of the data link for the delivery of both the data packet and the associated acknowledgment.

4. Packet Format and Link Control Protocols

The link level data packet format is shown in Figure 2. The data field is sandwiched between the packet header and trailing control fields. The header consists of the flag, "01111110," marking the beginning of a data packet, duplicate mark (DP), and the receiving process name field. The sending process name, packet sequence number and higher-level control information are considered here as parts of the

data field. The trailing control fields consist of the cyclic redundant check code (CRC), the acknowledgment/repeat request (ACK/RQ) field, and the occupied token "01111111",* marking the end of the data packet. The receiving process name and the data are supplied by the host. The other fields are generated by the sending RA.

We note that the data packet format is similar to that in HDLC. To achieve data transparency a zero is inserted following every occurrence of 5 contiguous 1's in the data steam between the flags and the occupied token as in HDLC. The flag and the token are the only control fields containing more than 5 1's and hence can be uniquely identified at link level. Before the zero insertion the data field is n x 16 bits long for some n between 0 and 255. The 16-bit CRC code specified by the generating polynomial $x^{16} + x^{12} + x^5 + 5$ is used for detecting errors in all bits between the flag and the ACK/RQ field.

A data packet is marked as a duplicate by the sending RA with its DP set to 1. A RA can check the first 16 bits (after zero deletion) following the flag to determine if the packet is intended for some local process and whether the data packet is a duplicate one. The last 8 bit field before the occupied token is the ACK/RQ field. When a data packet leaves the sending RA, its ACK/RQ field is reset to off to mean negative

*The bit pattern representing the occupied token is the same as that used to represent the control token. That a token is occupied (and, therefore, is not trapped by a RA which is waiting to obtain the token) is signified by this pattern following a matching flag.

acknowledgment and no repeat request. As the data packet passes through its front end, each RA on the ring may acknowledge whether the data packet is properly received by marking its comment in the ACK/RQ field. A RA sets the ACK field if the receiving process name matches the name of a local process name and if the data packet is copied and stored in its input buffer ready to be delivered to the local process. The RQ field is set when there is a process name match. However, either due to error detected in the data packet or due to input buffer overflow, the data packet is not correctly copied into the input buffer. Thus, the RA may request the data packet be retransmitted.

The operations of the sending RA is described by the flowchart in Figure 3. Before transmitting a data packet, the acknowledgment state of the sending RA and the number of retransmissions count are initially reset to zero. When the data packet is being transmitted for the first time, the duplicate mark is set to 0. As the data packet makes a round trip around the ring appropriate comments are collected in the ACK/RQ field from all RA's on the ring. By scanning the ACK field, the sending RA may determine whether the ACK field is set (meaning that some RA made a positive acknowledgment). If the ACK field is set, the acknowledgment state of the sending RA is set to 1. The repeat request field is set if any RA made a repeat request. The sending RA will immediately retransmit the data packet in this case. However, this time the DP bit is set to 1 to mark the data packet as a duplicate. If, on the other hand, the RQ field is found to be off when the data packet returns to the optical

i: transmission account

I: the number of maximum allowable transmission attempts

S: Acknowledgement State

DP: duplicate mark

ACK: Acknowledgement

RQ: Repeat Request

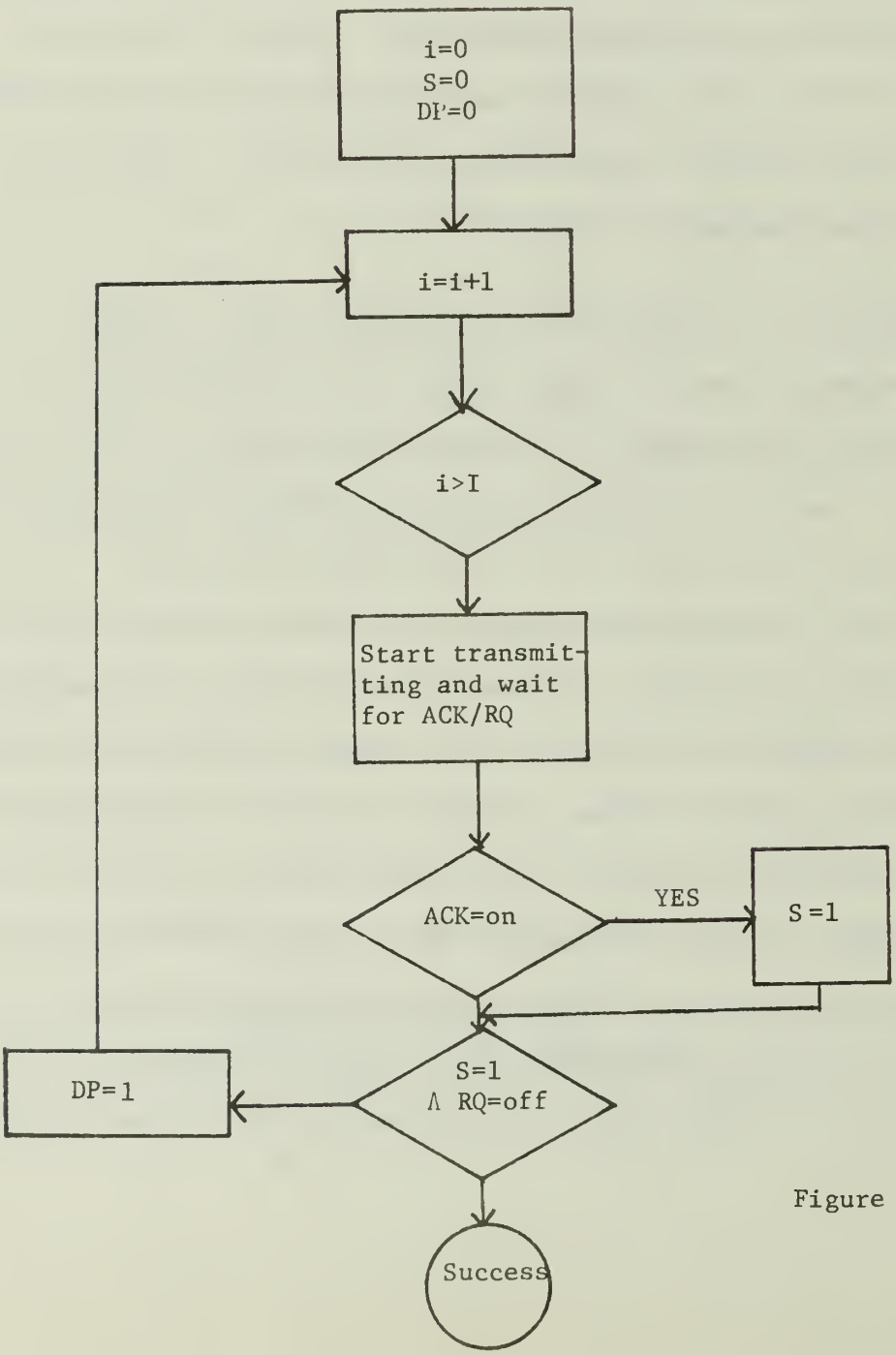
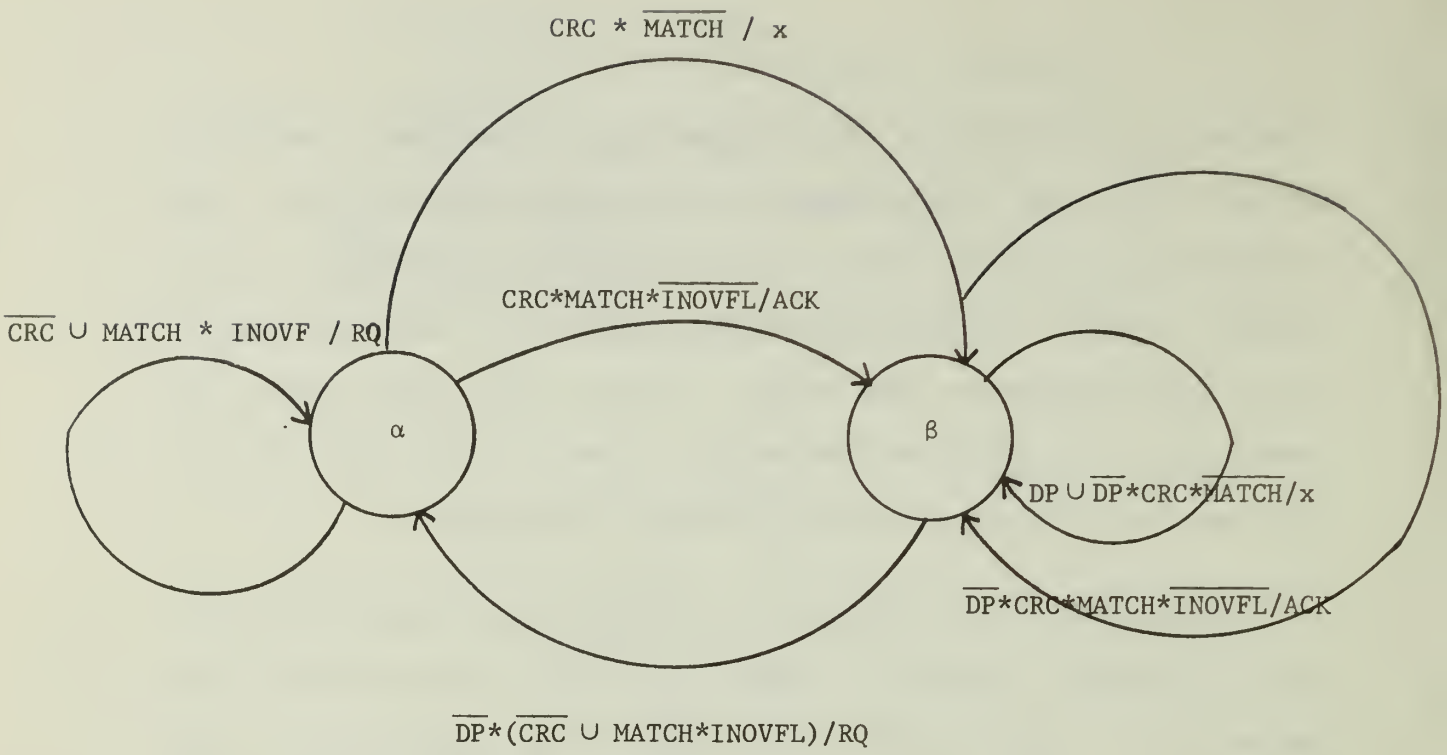


Figure 3

receiver in the sending RA, the transmission is considered completed. We note that the acknowledgment and the repeat request fields in the returned data packet not set by any RA will be interpreted by the sending RA that the receiving process name does not match the name of any active processes on the ring. In this case, the sending RA immediately retransmits the data packet. Since this data packet has not been received by any RA, it is not marked as a duplicate.

The operation of a RA which is not transmitting is described by the state transition diagram in Figure 4. Such a RA is in one of two states, α or β. When the DP in a data packet is 0, any RA may copy the data packet and make comment in the ACK/RQ field. Once a RA receives a data packet and stores it in an input buffer, it writes a positive acknowledgment in the ACK/RQ field of the data packet and enters state β. As this data packet reaches the sending RA, its acknowledgment state will be set to 1. Hence, during subsequent retransmission of the data packet, the DP is set to 1. While it is in state β, the RA is inhibited to make comment in any data packet marked as duplicate.

A RA enters state α when it makes a repeat request comment in the RQ field of the last data packet passing through its front-end window. When the duplicate mark in the data packet is set to 1, RA copies the data packet into its input buffer and makes positive acknowledgment in the ACK/RQ field only if it is in state α. Thus, reception of duplicate packets is prevented except in the relatively rare cases when noise



Legend: A/B

A: condition

CRC: CRC error code o.k.

DP: duplicate mark

MATCH: receiving process name of the data packet matches with that of some local process.

INOVFL: input buffer overflow

B: action

ACK: positive acknowledgement

RQ: repeat request

x: no comment

Figure 4

causes messages to be garbled on more than one link in the network.

To summarize, a RA which is not transmitting will set the ACK field of the data packet passing through its front-end window and thus make a positive acknowledgment of its reception if (1) it is in α state, or it is in β state, but the DP of the data packet is 0, (2) the receiving process name in the data packet matches some process name in the receiving RA, (3) there are free input buffers, and (4) the CRC check detected no error in the data packet. Similarly, it will make a repeat request if it is in state α, or it is in state β, the DP of data packet is 0, and one of the following conditions is true: (1) the CRC check found the data packet to be errorous, or (2) there is no free input buffer and the receiving process name of the data packet matches with that of some local process.

We note that there is no need to initialize the RA to be in α or β state. Being self-synchronized, the RA should function correctly even if some RA's are in state α and some RA's are in state β at the time when the transmission of any data packet commences.

4. Error Recovery

In the two nodes that have been implemented to date, error recovery hardware is not included. Because of the limited knowledge in the failure characteristics of the type of networks such as ILLINET, it was decided to postpone the implementation of these hardwares. Instead,

network error recovery functions are carried out by the hosts. However, time-out and interrupt circuits are included in each of the RA's for the detection of malfunctions in the RA or networks. For example, when a RA has a data packet to be sent but has waited for a long time for the control token, an interrupt is sent to the host when a preset time-out period expires to alert the host possible network malfunctions and invoke recovery procedure. Similarly, if after a RA caught the control token and transmitted a data packet but the occupied token at the end of the data packet does not return after a maximum loop delay or if the transmission lasted too long a period of time, appropriate interrupt signals are sent to the host. Status bits within the RA are provided to aid the host in its diagnosis to pinpoint the cause of network malfunction. The input buffer and output buffer memory modules are completely independent. Therefore, it is possible to support echo transmission mode. In this mode, a sending RA stores the data packet transmitted from its own output buffer when the packet returns from the network. It is also possible for a host to separate the RA from the network. In this case, a data packet may be transmitted directly from the output buffer to the input buffer of the RA. Thus, individual RA's can be tested independently making isolation of malfunctioned RA a relatively easy task.

Hardware for recovery from error conditions involving the token is included in our design. Under normal operating conditions there is only one control token circulating around the ring. Failure or transient

noises in both RA's or the link may cause the token to be lost or duplicated. We refer to these conditions as no token or duplicated token, respectively. Clearly, the no token condition exists when the network is turned on initially.

To explain how the no token or duplicated token conditions are to be handled in ILLINET, let us discuss the error conditions that can occur in ILLINET. As described above, all RA's monitor the data stream on the ring as it passes by their 16-bit front-end windows. Data streams arriving at the optical receiver in a RA is not relayed to the optical transmitter unless this data stream represents the access token or when it is preceded by a flag and the flag is detected by the RA. At the end of the data packet, the last remaining 16-bit of data in the front-end window are delivered to the optical transmitter for transmission only when the occupied token marking the end of the data packet is detected. Hence, any data stream with no leading flag and occupied token is blocked by the front-end of some RA. A data stream with a leading flag but no occupied token is truncated by 16-bits after passing through each RA until the data packet disappears. A data stream containing occupied token but no leading flag becomes a control token instead. Thus, the continuous circulation of random or broken data packets left on the ring due to failures in the sending RA or intermittant noise is prevented. "Garage collection" in this case is not required.

In the sending RA, the data path between the optical receiver and transmitter is normally open during the transmission of a data packet. This data packet is removed from the ring when it returns. If for some reason the data path within the sending RA is closed when the data packet returns to the sending RA, it will be left circulating on the ring. We note that this error condition is a serious one. If the duplicate mark in the packet is not set and if the receiving process name matches the names of some active process on the ring, the input buffers in the RA serving these processes will eventually overflow since the data packet will be copied by these RA's each time it passes by their front end. In this case, a RA monitoring the network will see well-formatted data packets pass by even though the no token condition exists. Since the sending process name and packet sequence number are considered as parts of data and not monitored by the RA's, this type of no token condition can be detected either by the receiving hosts after the received data packets have been examined or by a RA after waiting for some access token for a period of time longer than the maximum access delay on the ring. If in a N-node ring with loops delay L the maximum packet length is T seconds, and each RA is allowed to transmit k times before freeing the token, the maximum access delay is roughly (N-1)(k)(L+T). (For example, in a 6 node, 1 km ring network, the length of this period is approximately 10 μsec. with k = 16.) Fortunately, we believe that this type of no token condition rarely occurs in ILLINET. When it does occur, it is handled as follows: when a RA observes data

stream but no control token passes by its front-end window for a period of time longer than its estimated maximum access delay, it opens the data path between the optical receiver and transmitter. Thus, it removes the "garbage" from the ring. However, the no token condition persists.

The no token condition can be detected easily in the case when there is no data stream circulating on the ring. In this case, a RA can decide that there is no token on the ring after one maximum loop delay. (In our previous example, this time is roughly 7 μsec.) This type of no token condition is handled in the following manner. When a RA observes no data stream in the ring and there is no access token passing by its front end for a period longer than one maximum loop delay, it will enter a time-out period and continue to monitor the activities on the ring. If when its time-out expires and no token is observed, it will insert a token on the ring. By making the differences between the time-out periods of the different RA's equal to or longer than one loop delay, we are assured that once such a no token condition occurs, a token will be generated in a reasonably short time. Moreover, only one token will be generated in most cases.

The duplicate token detection scheme is designed for the general case when the exact loop delay is not known or may be variable. In this case, the duplicate tokens can be detected reliably at the host level. That there are more than one RA transmitting data packets at the same

time can be detected by the sending host by examining the sending process name and packet sequence number in the data field of the packets arriving at the optical receiver of its serving RA. However, the need of the host intervention will undoubtedly significantly lower the network throughput. Alternately, we may require that the sending process name be placed in the first 16 bits of the data field. The sending RA can, therefore, determine whether the packet arriving at the optical receiver is the same one sent on the ring by itself. When the received data packet is found to be from another RA, the sending RA can conclude that duplicate token conditions exist. Again, by removing all data streams arriving at its receiver, a sending RA will delete all tokens from the ring.

5. Hardware Structure

The hardware structure of the RA's already implemented is described by the block diagram shown in Figure 5. To satisfy the different speed requirements of the different functional blocks of the RA at a minimum cost and complexity, it is implemented in ECL, STTL, and LSTTL. A RA consists of three PC boards, the front-end, memory module and retransmission control logic, and RA-to-host interface.

The front end contains the transmitting and receiving logics. Between the optical receiver and transmitter, there is a shift register which serves as a delay buffer. The RA may hold up the incoming data

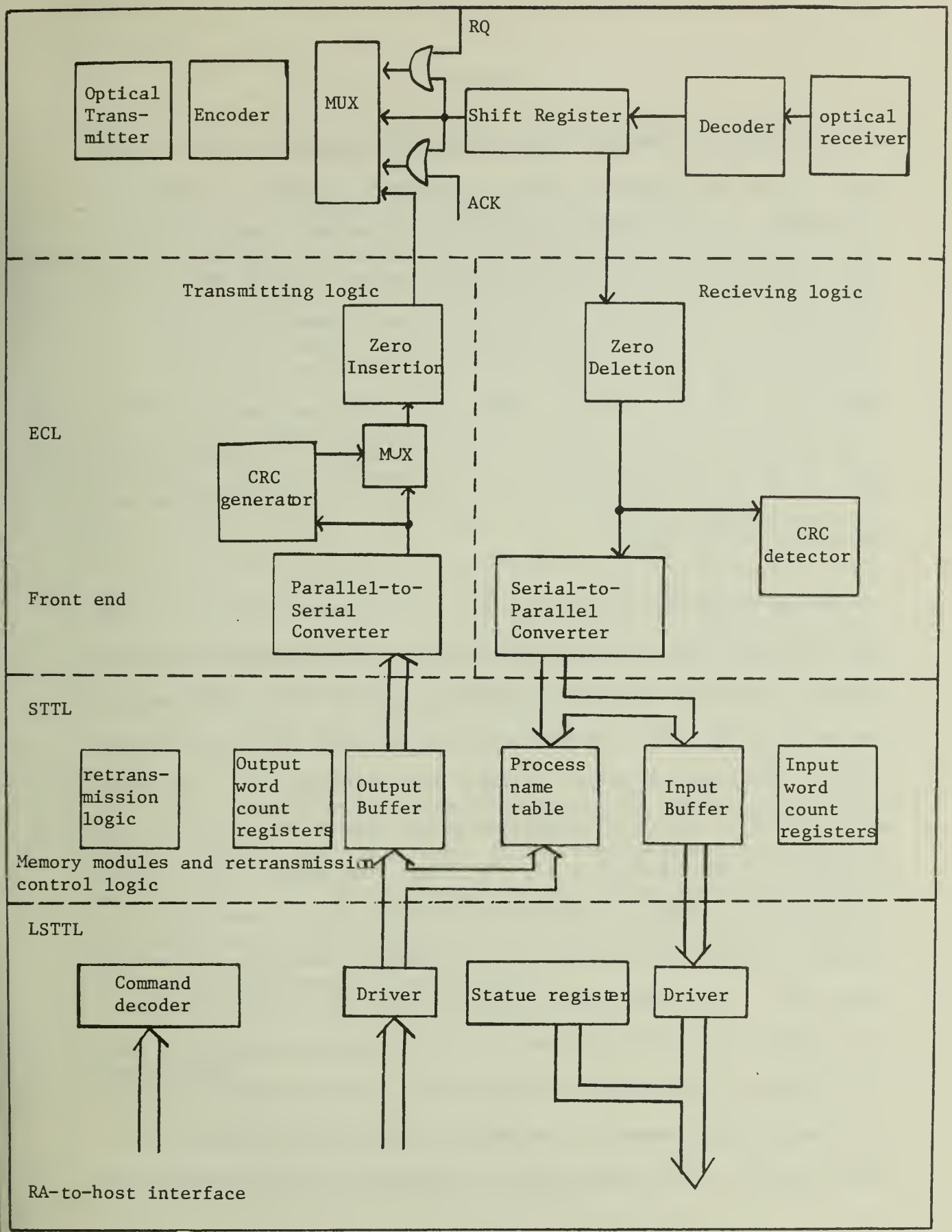

RQ
Optical Trans-mitter
Encoder
MUX
Shift Register
Decoder
optical receiver
ACK
Transmitting logic
Recieving logic
Zero Insertion
Zero Deletion
ECL
MUX
CRC generator
CRC detector
Front end
Parallel-to-Serial Converter
Serial-to-Parallel Converter
STTL
retrans-mission logic
Output word count registers
Output Buffer
Process name table
Input Buffer
Input word count registers
Memory modules and retransmission control logic
LSTTL
Command decoder
Driver
Statue register
Driver
RA-to-host interface

Figure 5

stream, scan and process the contents of the various fields as they appear in the shift register. Here, appropriate comment is generated and inserted in the ACK/RQ field of the data packet. Then the last 16 bits containing the ACK/RQ field and the token are shifted out to the transmitter. The major functions of the transmitting logic are parallel-to-serial data conversion, zero insertion CRC error code generation, and data packet formatting. The major functions of the receiving logic are zero deletion, serial-to-parallel conversion and CRC check. All these operations are carried out bit-serially and are implemented in ECL logic.

Within the memory modules and retransmission logic, the are input and output buffers, process name table, and retransmission control circuits. The buffer memory are segmented into 256 16-bit word pages. Sixteen pages are used as input buffers and two pages are used as output buffers. The input and output buffers are organized as independent modules, each is capable of supporting either read or write operation at 32 Mbits/sec. Upon detection of the flag, the input buffer write operation is initiated. If at the time data is being transferred from the input buffer to the host, this transfer operation is halted temporarily. The input buffer write operation will be terminated when the occupied token marking the end of the data packet is detected, when the receiving process name in the data packet does not match any names in the process name table, or when the duplicate mark is found to be set indicating that the date packet is already copied by the RA. Any

temporarily halted memory transfer operation will then be resumed.

There are two output buffers in the output modules to allow the process of waiting for network access and data transfer from the host to be carried out concurrently. A 32 Kxl memory module implemented with Intel 2147H3 is used to store receiving process names. (Currently, we are using only one chip containing 4 K in each RA.) The 15-bit receiving process name is used to address this RAM table. An output bit from the table being 1 indicate a match of the receiving process name with some local process name in the table. Thus, the process of checking receiving process name match can be carried out in 55 nsec. The table can be dynamically updated by the host within 500 nsec. All buffer memory operations, receiving process name checking and updating, and retransmission control are carried out at word level and are implemented in STTL logic.

Finally, the RA-to-host interface contains the command decoder, RA status registers and interfaces to and from buffer memory modules. These circuits allow the RA to appear to the host as a peripheral device and can be easily linked to the host via a DMA interface. This portion of the RA is implemented in the TTL logic.

Acknowledgment

The authors wish to thank Cyrus Weise, Kurt Horton, Izumi Suwa for suggestions and help in the design and implementation of ILLINET.

References

[1] Metcalfe, R.M. and D.R. Boggs, Ethernet: distributed packet switching for local computer network, CACM 19, 7, July 1976, 395-404.

[2] Frazer, A.G., Spider--an experimental data communication system, Proceedings International Communications Conference, 21F-1-10, CACM.

[3] Pogram, K.T. and D.P. Reed, The MIT laboratory for computer science network, Local Area Networking NBS Special Publication 500-31, April 1978, 22-23.

[4] Weber, H., D. Baum and R. Popescu-Zelltin, ESA--an evolutionary system architecture for a distributed data base management system, Proceedings of Berkeley Conference on Distributed Processing, 1979.

[5] Farber, D.J., A ring network, Datamation, February 1975, 44-46.

[6] Burton, H.O. and D.O. Sullivan, Error and error control, Proceedings of the IEEE 60, 1972.

[7] Mockopetris, P., Design consideration and implementation of ARPA LNI nametable, University of California, Dept. of Information and Computer Science, Technical Report 92, Irvine, CA, April 1978.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-80-1035	2.	3. Recipient's Accession No.
4. Title and Subtitle ILLINET--A 32 Mbits/sec. Local Area Network*			5. Report Date October 1980 6.
7. Author(s) W.Y. Cheng, S. Ray, R. Kolstad, J. Luhukay, R. Campbell, J.W-S. Liu			8. Performing Organization Rept. No.
9. Performing Organization Name and Address Department of Computer Science University of Illinois 222 Digital Computer Lab Urbana, IL 61801			10. Project/Task/Work Unit No. 11. Contract/Grant No. NSF MCS 79-06945
12. Sponsoring Organization Name and Address National Science Foundation Washington, D.C.			13. Type of Report & Period Covered 14.

15. Supplementary Notes

16. Abstracts

ILLINET is a fiber-optical ring network designed to provide wide band linkages between host computers for the purpose of facilitating file transfers at speeds near those of fast I/O devices in the hosts. Its structure is similar to the Distributed Computing System. ILLINET will eventually connect several PDP-11's, a PRIME computer and a network of microcomputers. These computers are used in a variety of real-time and batch processing applications. Currently they are already interconnected via 9600 band lines in a star configuration to provide access to simple terminals. This paper describes the network architecture, control structure, and hardware configuration of ILLINET.

17. Key Words and Document Analysis. 17a. Descriptors

Local-Area network
Ring network

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 28
	20. Security Class (This Page UNCLASSIFIED	22. Price